Dorn et al. Algorithms for Molecular Biology 2011, 6:21
http://www.almob.org/content/6/1/21

AMB ALGORITHMS FOR MOLECULAR BIOLOGY

RESEARCH Open Access

Exploiting bounded signal flow for graph orientation based on cause–effect pairs

Britta Dorn[1], Falk Hüffner[2*], Dominikus Krüger[3], Rolf Niedermeier[2] and Johannes Uhlmann[2]

Abstract

Background: We consider the following problem: Given an undirected network and a set of sender–receiver pairs, direct all edges such that the maximum number of "signal flows" defined by the pairs can be routed respecting edge directions. This problem has applications in understanding protein interaction based cell regulation mechanisms. Since this problem is NP-hard, research so far concentrated on polynomial-time approximation algorithms and tractable special cases.

Results: We take the viewpoint of parameterized algorithmics and examine several parameters related to the maximum signal flow over vertices or edges. We provide several fixed-parameter tractability results, and in one case a sharp complexity dichotomy between a linear-time solvable case and a slightly more general NP-hard case. We examine the value of these parameters for several real-world network instances.

Conclusions: Several biologically relevant special cases of the NP-hard problem can be solved to optimality. In this way, parameterized analysis yields both deeper insight into the computational complexity and practical solving strategies.

Background

Current technologies [1] like two-hybrid screening can find protein interactions, leading to protein-protein interaction (PPI) networks, but cannot decide the direction of the interaction. This can be complemented by gene knock-out experiments which constitute a way to determine causal relations in these networks, thus providing additional information on possible directions of information flow in them [2]. Given a list of so-called cause–effect pairs, the challenge consists in deducing an orientation of the PPI network which takes into account the causal relations of as many of these pairs as possible. Medvedovsky et al. [3] formalize this in terms of a graph theoretical problem as follows.

Problem Formalization

Let $G = (V, E)$ be an undirected graph. An *orientation* $\vec{G}$ of G is a directed graph $\vec{G} = (V, \vec{E})$ obtained from G by replacing every undirected edge $\{u, v\} \in E$ by a directed one, i. e., either by $(u, v) \in \vec{E}$ or by $(v, u) \in \vec{E}$. Let $P \subseteq V \times V$ be a set of ordered source–target pairs, which we sometimes refer to as "signals". In order to distinguish pairs from edges or arcs, we use the notation $[a, b] \in P$ to denote the pair starting in a and ending in b. We say that a pair $[a, b] \in P$ is *satisfied* by a given orientation $\vec{G}$ if there exists a directed path from a to b in $\vec{G}$. The central problem considered in this work is to find an orientation of a given graph maximizing the number of satisfied pairs. As pointed out by Medvedovsky et al. [3], we can assume that the given graph is a tree: it is clearly optimal to orient the edges of a cycle to form a directed cycle, and hence one can repeatedly contract a cycle to a single vertex, obtaining a tree. Note that this process will always produce the same tree independent of the order of contractions, since two vertices will be merged eventually if and only if they are in the same bridge block, where a bridge block is a connected component of the graph that is obtained by deleting all bridges (edges whose deletion increases the number of connected components). Further, bridge blocks can be found in linear time [4,5]. Thus, formalized as a decision problem, Maximum Tree Orientation is defined as follows.

* Correspondence: falk.hueffner@tu-berlin.de
[2]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany
Full list of author information is available at the end of the article

Maximum Tree Orientation (MTO)

Input: An undirected tree T, a set P of ordered pairs of vertices of T, and an integer $k \geq 0$.

Question: Is there an orientation of T such that at most k pairs in P are not satisfied?

We also consider the weighted version, called Weighted Maximum Tree Orientation (W-MTO), where every pair $[a, b] \in P$ is associated with a rational weight $\omega([a, b]) \geq 0$, and the goal is to maximize the sum of weights of the satisfied pairs. Throughout this work, n denotes the number of vertices in the given MTO instance, if not stated otherwise.

As sketched before, MTO is motivated from the inference of causal relations in biological networks [6,7] such as PPI networks, but it also has applications in the context of communication networks, where several one-way connection request pairs are given. Since each link between two network nodes can only be used in one direction, one has to orient the links in such a way that as many communication requests as possible can be fulfilled.

Previous Work

MTO was introduced by Medvedovsky et al. [3]; they showed that the problem is NP-complete even when the underlying tree is a star (that is, a diameter-two tree) or a tree with maximum vertex degree three. Moreover, they provided a cubic-time algorithm for MTO restricted to paths. Seeing MTO as the task to maximize the number of satisfied pairs, Medvedovsky et al. also provided polynomial-time approximation algorithms with approximation factor 1/4 in the case of stars and $O(1/\log n)$ in the case of general n-vertex trees. The latter approximation factor was recently improved to $O(\log \log n/\log n)$ by Gamzu et al. [8], who furthermore extended the studies of MTO to "mixed graphs" where some of the edges are already oriented based on causal relations known in advance. Besides these theoretical investigations, Medvedovsky et al. [3] also provided some experimental results based on a yeast PPI network and some synthetic data. Silverbush et al. [9] recently formulated a polynomial-size integer linear program for the generalization of mixed graphs and did some experiments with it. Also recently, Gitter et al. [10] considered graph orientation with the objective of maximizing the weight of all satisfied paths between sources and targets with length at most some constant k. They used approximation algorithms to discover pathways in biological networks. In an earlier work, Hakimi et al. [11] studied the special case of MTO where the list of pairs to be satisfied contains *all* possible pairs; they developed a quadratic-time algorithm for this case.

Our Contributions

We mainly continue and complement previous work on MTO [3,8] by starting a parameterized and multivariate complexity analysis of MTO. That is, we try to better understand the border between tractable and intractable cases of MTO while sticking to optimal (instead of approximate) solutions. In particular, our focus is on the "amount of signal flow" over vertices and edges, respectively, and how this influences the computational complexity of MTO.

- We show that W-MTO can be solved in $O(2^{m_v} \cdot |P| + n^4)$ time on an n-vertex tree, where m_v denotes the maximum number of connecting paths (one-to-one corresponding to the input vertex pairs) over any tree vertex. In other words, W-MTO is fixed-parameter tractable with respect to the parameter m_v.
- We introduce the concept of cross pairs and show that cross-pair-free instances of W-MTO can be solved in quadratic time, as a corollary also improving the cubic-time algorithm of Medvedovsky et al. [3] for MTO on paths to quadratic time.
- We additionally show that W-MTO is fixed-parameter tractable with respect to the parameter q_v which is the maximum number of cross pairs over any vertex; namely, it can be solved in $O(2^{q_v} \cdot n^2 \cdot q_v)$ time.
- Shifting the focus from "maximum vertex signal flow" to "maximum edge signal flow", we show a sharp complexity dichotomy: W-MTO can be solved in linear time if no tree edge has to carry more than two signals, but if this maximum edge signal flow is three, MTO already becomes NP-hard.
- Finally, we briefly discuss some practical aspects of exactly solving the so far very few considered real-world instances and conclude that these can be already solved to optimality within milliseconds (via at least three different strategies). However, we also make the point that with the future availability of further real-world data, our new algorithms could be of significant practical relevance beyond so far known or straightforward approaches.

Preliminaries, Basic Facts, and Simple Observations

For ease of presentation, for a W-MTO instance (T, P, ω), we always assume that $\omega([s, t]) = 0$ for all pairs $s, t \in V$ with $[s, t] \notin P$. Moreover, subsequently mostly referring to MTO, the presented concepts and definitions clearly apply to W-MTO as well. Note that in a tree $T = (V, E)$, for each ordered pair $[a, b]$ of vertices, there exists a uniquely determined path connecting these vertices. We will therefore often write *the path defined by the pair* $[a, b]$ when we refer to the unique path in the tree starting in vertex a and ending in vertex

b, or talk about pairs and paths interchangeably. Sometimes, we also talk about paths in the tree which do not necessarily correspond to pairs. We denote the undirected path connecting vertices v and w in T by $\text{path}_T(v, w)$. Moreover, $P_v := \{[s, t] \in P \mid v \in V(\text{path}_T(s, t))\}$ denotes the set of paths *passing through a vertex* v (note that this includes paths of which v is an endpoint). An MTO instance is called *rooted* if the underlying tree T is rooted. In a rooted tree $T = (V, E)$, if vertex $a \in V$ is an ancestor of vertex $b \in V$, then we use the notation $a \prec b$. The subtree of T rooted at $v \in V$ is denoted T_v.

Let $(T = (V, E), P)$ be an MTO instance, and let $x, y \in P$ be two pairs. We say that x *conflicts with* y if there exists no orientation of T for which both x and y are satisfied. From an n-vertex MTO instance, we build a so called *conflict graph* in which each vertex corresponds to an input pair of the MTO instance, and where there is an edge between two pairs if and only if they conflict with each other. More formally, given an MTO instance $(T = (V, E), P)$, the corresponding conflict graph $G_c(T, P)$ is defined $G_c(T, P) := (P, E_c)$ where $E_c := \{\{u, v\} \mid u, v \in P \wedge u \text{ conflicts with } v\}$.

The computation of the conflict graph can be done in $\Theta(n^4)$ time. It clearly cannot be done faster, because up to $O(n^4)$ conflicts are possible. To achieve the desired bound, we thus need to decide in constant time whether two pairs conflict with each other. This is done using an appropriate data structure and two simple observations: First, in a rooted tree, least common ancestors (LCAs) can be calculated in constant time after some linear time preprocessing [12]. Second, two pairs are in conflict if and only if their paths run in different directions through an edge incident on the lower one of the two LCAs of the two pairs. Clearly, for an orientation of (T, P), in G_c there are no edges (that is, conflicts) between the vertices corresponding to the satisfied source–target pairs, and hence the vertices corresponding to the non-satisfied source–target pairs form a vertex cover for G_c, that is, a vertex set $V' \subseteq P$ such that for every edge $e \in E_c$ at least one endpoint of e is in V'. This yields the following useful observation.

Proposition 1. *Finding a minimum-weight vertex cover in the conflict graph $G_c(T, P)$ one-to-one corresponds to determining a minimum-weight set of pairs that cannot be satisfied in (T, P).*

It is generally assumed that the fact that a problem is NP-hard implies that there is no algorithm that finds an optimal solution and has running time bounded by a polynomial of the size of the input. Parameterized complexity is a *two-dimensional* framework for the analysis of computational complexity [13-15]. One dimension is the input size n, and the other one is the *parameter* (usually a positive integer). A problem is called *fixed-parameter tractable* (fpt) with respect to a parameter x if it can be solved in $f(x) \cdot n^{O(1)}$ time, where f is a computable function only depending on x. If a problem is fixed-parameter tractable with respect to x, we can hope for efficient optimal solutions as long as the parameter is not too large. Due to Proposition 1 we can immediately conclude that MTO and W-MTO are fixed-parameter tractable with respect to the parameter "number of pairs" p, since the conflict graph has p vertices and we can find a minimum-weight vertex cover by trying all possibilities in $2^p \cdot n^{O(1)}$ time. Further, since minimum-weight vertex covers can be found in $O(1.379^k + kn)$ time [16], we have fixed-parameter tractability with respect to the parameter "number of unsatisfied pairs", and if all weights are at least one, also with respect to the parameter "total weight of unsatisfied pairs".

Tree-decomposition-based algorithms have been successfully applied in the area of computational biology, for instance, in the context of structure–sequence alignment [17]. Informally speaking, the *treewidth* [15] measures the "tree-likeness" of a graph, and a tree decomposition is the "embedding" of a graph into a tree depicting the tree-like structure of the graph.

We recall the following definitions from literature [18]: A *tree decomposition* of a graph $G = (V, E)$ is a pair $\langle\{X_i | i \in I\}, \mathcal{T}\rangle$, where each X_i is a subset of V called *bag*, and $\mathcal{T} = (I, F)$ is a tree with node set I and edge set F. The following must hold:

1. $\bigcup_{i \in I} X_i = V$;
2. for every edge $\{u, v\} \in E$, there is an $i \in I$ such that $\{u, v\} \subseteq X_i$; and
3. for all $i, j, l \in I$, if j lies on the path between i and l in $\mathcal{T}$, then $X_i \cap X_l \subseteq X_j$.

The *width* of $\langle\{X_i | i \in I\}, \mathcal{T}\rangle$, is $\max\{|X_i| \mid i \in I\} - 1$. The *treewidth* of G is the minimum width over all tree decompositions of G.

Methods and Results

Bounded Signal Flow Over Vertices

In this subsection, we investigate how the vertex-wise structure of the source–target pairs influences the computational complexity of Maximum Tree Orientation. More specifically, first we consider the parameter m_v denoting the maximum number of source–target paths passing through a vertex. We show that MTO can be solved in $O(2^{m_v} \cdot |P| + n^4)$ time. In other words, MTO is fixed-parameter tractable with respect to the parameter m_v. Motivated by this positive result, we explore in more depth the structure of the source–target paths that pass through a vertex. To this end, we introduce the concept of "cross pairs" and show that for cross-pair-free instances MTO can be solved in $O(n^2)$ time. Informally speaking, an instance is cross-pair-free if the input tree can be rooted such that for each source–target pair one endpoint is an ancestor of the other one.

Then, for a rooted MTO instance a cross pair is a source–target pair such that none of its endpoints is the ancestor of the other endpoint. By refining the solving strategy for cross-pair-free instances, we show that Maximum Tree Orientation can be solved in $O(2^{q_v} \cdot n^2 \cdot q_v)$ time, where q_v denotes the maximum number of cross pairs passing through a vertex.

All algorithms in this subsection are based on dynamic programming, and, hence, since source–target pair weights can easily be incorporated, extend to W-MTO.

Parameter "Maximum Number of Pairs Per Vertex"

Here, we show that W-MTO is fixed-parameter tractable for the parameter m_v denoting the maximum number of source–target pairs passing through a vertex. To this end, we prove in Theorem 1 that we can construct in polynomial time a tree decomposition of the conflict graph of treewidth at most m_v. Recall that (weighted) MTO is equivalent to (weighted) Vertex Cover on the conflict graph (see Proposition 1). Thus, the running time follows by the fact that (weighted) Vertex Cover can be solved in $O(2^{tw}n)$ time, given a tree decomposition of width tw [15]. □

Theorem 1. *On n-vertex trees,* Weighted Maximum Tree Orientation *is solvable in* $O(2^{m_v} \cdot |P| + n^4)$ *time, where* m_v *denotes the maximum number of source–target pairs passing through a vertex.*

Proof. First, we show how to construct a tree decomposition of width m_v of the conflict graph in polynomial time. Let $(T = (V, E), P)$ denote an MTO instance and let $G_c = (P, E_c)$ denote the associated conflict graph. The basic idea is that we can use T as the underlying tree of a tree decomposition of $G_c = (P, E_c)$. More specifically, the tree decomposition is given by $\langle\{P_v \mid v \in V\}, T\rangle$ for all $v \in V$. Recall that P_v denotes the set of source–target pairs passing through v. Observe that each vertex $p \in P$ of the conflict graph appears exactly in the bags X_v for all $v \in V$ ($\text{path}_T(p)$). Moreover, note that if two source–target pairs $p = [s, t]$ and $p' = [s', t']$ are in conflict (and hence are adjacent in the conflict graph), then $\text{path}_T(s, t)$ and $\text{path}_T(s', t')$ have at least one edge and thus at least two vertices in common. Hence, every edge of the conflict graph is contained in at least one of the X_v's. Thus, all conditions of a tree decomposition are fulfilled. Moreover, the width of this tree decomposition is clearly m_v - 1. The conflict graph, the sets P_v, and the tree decomposition can be computed in $O(n^4)$ time. Thus, the overall running time follows by the fact that Weighted Vertex Cover can be solved in $O(2^{tw}|P|)$ time, given a tree decomposition of width tw of G_c [15].

Cross Pairs

In Theorem 1, we have shown that W-MTO is fixed-parameter tractable with respect to the parameter m_v. In the following, we will strengthen this result by showing that W-MTO is fixed-parameter tractable with respect to the parameter "number of a special type of source–target pairs (the so-called cross pairs) passing through a vertex". The idea is to identify a "trivial" (that is, polynomial-time solvable) special case of the problem and then to investigate instances that are close to these trivial instances, their closeness measured in terms of a certain parameter which is referred to as *distance from triviality* [19,20].

In the following, we will always consider *rooted* trees. Informally speaking, a cross-pair-free instance only contains source–target pairs whose corresponding paths are directed either towards the root or towards the leaves, but do not change their direction. Cross-pair-free instances of W-MTO are of special interest since they constitute our "trivial instances".

Definition 1. *Let* $(T = (V, E), P, \omega)$ *be an instance of W-MTO where T is a rooted tree. A source–target pair* $p = [a, b] \in P$ *is called cross pair if neither a is an ancestor of b nor b an ancestor of a. An instance of W-MTO is called cross-pair-free if T can be rooted such that P does not contain any cross pairs.*

Cross-pair-free Instances

Now, we devise a dynamic-programming-based algorithm solving W-MTO in quadratic time on cross-pair-free instances.

Theorem 2. *On n-vertex trees,* Weighted Maximum Tree Orientation *for cross-pair-free instances with given root can be solved in* $O(n^2)$ *time.*

Proof. **Algorithm**. We present a dynamic programming algorithm with quadratic running time solving a cross-pair-free W-MTO instance $(T = (V, E), P, \omega)$ with root r. For the presentation of the algorithm, we use the following notation. For $v \in V$, let T_v be the subtree of T rooted at v. For all $v, w \in V$ with $v \prec w$ (that is, v is an ancestor of w) let T_w^v denote the subtree of T induced by $V_w^v := V(T_w) \cup V(\text{path}_T(v, w))$.

For ease of presentation, let $V_w^w := V(T_w)$. Moreover, let $P_w^v := \{[s, t] \in P \mid s, t \in V_w^v\}$. That is, T_w^v is the tree consisting of the path $\text{path}_T(v, w)$ and the subtree T_w rooted at w, and P_w^v are the pairs with both endpoints in T_w^v. Finally, the *weight* of an orientation $\vec{T}_w^v$ of (T_w^v, P_w^v) is the sum of the weights of the pairs in P_w^v satisfied by $\vec{T}_w^v$.

The algorithm maintains an $n \times n$ dynamic programming table S, containing for each $v, w \in V$ with $v \prec w$ or $v = w$ the two entries $S(v, w)$ and $S(w, v)$. The goal of the dynamic programming procedure is to fill S in accordance with the following definition.

For all $v, w \in V$ with $v \prec w$, entry $S(v, w)$ is the maximum weight of an orientation of (T_w^v, P_w^v) among all orientations of (T_w^v, P_w^v) orienting the path between v and w from v to w (that is, away from the root).

Analogously, $S(w, v)$ is the maximum weight of an orientation of (T_w^v, P_w^v) among all orientations of (T_w^v, P_w^v) orienting the path between v and w from w to v (that is, towards the root). Note that in the case $v = w$ we have that $S(v, v)$ is the weight of an optimal orientation of the subtree rooted at v.

Next, we describe how our algorithm computes the entries of S in accordance with this definition. The weight of an optimal orientation of (T, P) can then be found in $S(r, r)$.

To compute the entries of S, visit all vertices $w \in V$ in a bottom-up traversal. Then, for each w consider all vertices $v \in V$ with $v = w$ or $v \prec w$ and set (omit the sum if w is a leaf):

$$S(v,w) := A(v,w) + \sum_{u \text{ is a child of } w} \max\left\{S(u,w), S(v,u) - A(v,w)\right\}, \quad (1)$$

$$S(w,v) := A(w,v) + \sum_{u \text{ is a child of } w} \max\left\{S(w,u), S(u,v) - A(w,v)\right\}. \quad (2)$$

Herein, $A(w, v)$ denotes the sum of the weights of the source–target pairs with both endpoints on $\text{path}_T(v, w)$ that are satisfied when orienting the path between v and w from v to w, that is,

$$A(v,w) := \omega(\{[s,t] \in P \mid s,t \in V(\text{path}_T(v,w)) \wedge s \prec t\}). \quad (3)$$

Analogously, $A(w, v) := \omega$ ({[s, $t \in P \mid s, t \in V$ ($\text{path}_T(v$, w)) $\wedge$ $t \prec s$}). Moreover, for ease of presentation we assume that $A(v, v) = 0$.

Correctness. For the correctness of the algorithm note the following. For a leaf w and an ancestor v of w, the tree T_w^v is identical to the path $\text{path}_T(v, w)$. Hence, the sum of the weights of pairs that can be satisfied by orienting the path either from v to w or from w to v is $A(v, w)$ and $A(w, v)$, respectively. Next, consider the case that w is an inner vertex and let v be an ancestor of w. Moreover, let $u_1, \cdots, u_\ell$ denote the children of w. We argue that the maximum weight of an orientation of (T_w^v, P_w^v) orienting the edges on $\text{path}_T(v, w)$ towards w equals

$$A(v,w) + \sum_{i=1}^{\ell} \max\left\{S(u_i,w), S(v,u_i) - A(v,w)\right\}, \quad (4)$$

and, hence, $S(v, w)$ is computed correctly. To this end, consider a maximum-weight orientation $\vec{T}_w^v$ of (T_w^v, P_w^v) orienting the edges on $\text{path}_T(v, w)$ towards w. If, for a child u_i, $\vec{T}_w^v$ contains the arc (u_i, w), then the contribution of the source–target pairs in P_w^v with at least one endpoint in T_{u_i} to the weight of $\vec{T}_w^v$ is $S(u_i, w)$; note that no source–target pair of P_w^v with exactly one endpoint in $T_{u_i}^w$ is satisfied by $\vec{T}_w^v$, and thus the contribution of these pairs is S (u_i, w) (a smaller contribution would contradict the optimality of $\vec{T}_w^v$). Moreover, if for a child u_i the oriented tree $\vec{T}_w^v$ contains the arc (w, u_i), then it follows by a similar argument that the contribution of the paths in P_w^v with at least one endpoint in $V(T_{u_i})$ is $S(v, u_i)$ - $A(v, w)$. The only difference is that the contribution of the source–target pairs with both endpoints in V ($\text{path}_T(v, w)$) is already considered in the above formula, and, hence, must be subtracted from $S(v, u_i)$.

Running time. For the running time bound, we show that A can be computed in $O(n^2)$ time in a preprocessing step. Then, the overall running time is clearly bounded by

$$O\left(\sum_{v \in V}\sum_{w \in V} \deg_T(w)\right) = O\left(\sum_{v \in V} n\right) = O(n^2), \quad (5)$$

since $\sum_{w \in V} \deg_T(w) = 2(n - 1)$ in trees. Clearly for v, $w \in V$ with $v \prec w$ the matrix entries $A(v, w)$ and $A(w, v)$ can be computed by setting

$$A(v,w) := \omega([v,w]) + A(v,\gamma) + A(x,w) - A(x,\gamma) \quad (6)$$

and

$$A(w,v) := \omega([w,v]) + A(w,x) + A(\gamma,v) - A(\gamma,x), \quad (7)$$

where x is the neighbor of v and y is the neighbor of w on $\text{path}_T(v, w)$. This assumes, however, that all the entries of A for pairs with distance l - 1 are known before computing the entries of the pairs with distance l. This can be ensured by using a queue (first-in-first-out data structure) as follows. For the computation of A, first, for all edges $\{v, w\} \in E$ with $v \prec w$ set $A(v, w) := \omega([v, w])$ and A (w, v) := $\omega([w, v])$ (let $\omega([s, t]) := 0$ if $[s, t] \notin P$) and append the pair (v, w) at the tail of the queue. After each edge has been processed, proceed as follows. Until the queue is empty, let (x, w) denote the next element at the head of the queue and let v denote the parent of x and let y denote the parent of w in T (if $x = r$ remove (x, w) from the head position of the queue and do nothing else). It is easy to verify that the entries for the pairs (x, w) and (v, y) have already been computed and, hence, can be used to compute $A(v, w)$ and A(w, v) as described above. Finally, we remove (x, w) from the head position of the queue and append (v, w) at the tail of the queue. Clearly, for every pair v, w of vertices with $v \prec w$, we need a constant number of operations to compute the two table entries $A(v, w)$ and $A(w, v)$, resulting in an overall running time bound of $O(n^2)$. □

Note that if the root of a cross-pair-free W-MTO instance is not known, it can be calculated in $O(n|P|)$ time by trying all roots and then checking for each pair if the least common ancestor is one of the two endpoints.

As an immediate consequence of Theorem 2, we can improve the cubic-time algorithm for MTO on paths by Medvedovsky et al. [3] to quadratic time. Herein, we use that every path rooted at one of its endpoints results in a cross-pair-free instance of MTO.

Corollary 1

Weighted Maximum Tree Orientation *on n-vertex paths can be solved in* $O(n^2)$ *time.*

Parameter "Maximum Number of Cross Pairs Passing Through a Vertex"

Next, we show that W-MTO is fixed-parameter tractable with respect to the parameter q_v by extending the dynamic programming algorithm for cross-pair-free instances. Formally, q_v is defined as follows. For a rooted W-MTO instance $(T = (V, E), P)$ with root r, let Q denote the set of cross pairs. Moreover, for $v \in V$ let $Q_v := P_v \cap Q$ be the set of cross pairs passing through v. With respect to the root r the maximum number $q_v(r)$ of cross pairs passing through a vertex is given by $\max_{v \in V} |Q_v|$. Then, q_v is the minimum value of $q_v(r)$ over all possible choices r to root T.

Theorem 3. *On n-vertex trees,* Weighted Maximum Tree Orientation *with given root can be solved in* $O(2^{q_v} \cdot q_v \cdot n^2)$ *time, where* q_v *denotes the maximum number of cross pairs passing through a vertex.*

Proof. The basic idea of the algorithm is to incorporate the cross pairs by trying for every vertex all possibilities to satisfy the cross pairs passing through this vertex. To this end, we extend the matrix S by an additional dimension. As a consequence, the dynamic programming update step becomes significantly more intricate.

Let $(T = (V, E), P, \omega)$ be a rooted W-MTO instance with root r. For the presentation of the algorithm we use the same notation as in the proof of Theorem 2. In addition, we employ the following definitions.

Let $w \in V$. A possibility to satisfy the cross pairs in Q_w is represented by a coloring $c_w : Q_w \to \{0, 1\}$, meaning that a cross pair $q \in Q_w$ must be satisfied iff $c_w(q) = 1$. Let $\mathcal{C}_w$ denote the set of all 0/1-colorings of Q_w. Note that $|\mathcal{C}_w| = 2^{|Q_w|}$. To incorporate the cross pairs, for every $v, w \in V$ with $v \prec w$ or $v = w$ and for every coloring c_w, the dynamic programming table S contains two entries $S(v, w, c_w)$ and $S(w, v, c_w)$. Informally speaking, $S(v, w, c_w)$ denotes the maximum weight of an orientation of T_w^v under the assumption that all cross pairs $q \in Q_v$ with $c_w(q) = 1$ are "satisfiable" and the edges in $\text{path}_T(v, w)$ are oriented from v towards w. Entry $S(w, v, c_w)$ is defined analogously, but here we assume that the edges in $\text{path}_T(v, w)$ are oriented from w towards v. For a precise description, we use the following notation.

Clearly, we are interested only in colorings c_w of Q_w such that any two cross pairs $q, q' \in Q_w$ with $c_w(q) = c_w(q') = 1$ are not in conflict. We call such a coloring *locally feasible*. Moreover, we extend the notion "feasible" as follows. As informally described above, we distinguish the two cases that the edges on $\text{path}_T(v, w)$ are oriented

(1) from v to w (for entry $S(v, w, c_w)$), or

(2) from w to v (for entry $S(w, v, c_w)$).

For the case (1), a locally feasible coloring c_w is called *feasible* if for each cross pair $[s, t] \in Q_w$ with $c_w([s, t]) = 1$ orienting the edges on $\text{path}_T(v, w)$ from v to w and the edges on $\text{path}_T(s, t)$ from s to t is simultaneously possible. Analogously, a locally feasible coloring c_w is feasible for case (2) when orienting the edges on $\text{path}_T(s, t)$ from s to t does not contradict the orientation of the edges on $\text{path}_T(v, w)$ from w to v.

For a coloring c_w of Q_w, we must ensure that in the considered orientations of (T_w^v, P_w^v) all cross pairs $q \in Q_w$ with $c_w(q) = 1$ are satisfiable. Therefore, we call an orientation of (T_w^v, P_w^v) *consistent with a cross pair* $[s, t] \in Q_w$ (note that $[s, t] \in P \setminus P_w^v$ is allowed) if the common edges of T_w^v and $\text{path}_T(s, t)$ are oriented from s to t. Finally, we call an orientation of (T_w^v, P_w^v) *consistent with a coloring* c_w of Q_w if the orientation is consistent with each cross pair $q \in Q_w$ with $c_w(q) = 1$.

With these notations, we can formally define the meaning of the entries of S. For every two vertices $v, w \in V$ with $v \prec w$ or $v = w$ and for every 0/1-coloring $c_w \in Q_w$ the entry $S(v, w, c_w)$ is $-\infty$ if the coloring c_w is not feasible for case (1). Otherwise, $S(v, w, c_w)$ denotes the maximum weight of an orientation of (T_w^v, P_w^v) among all orientations of (T_w^v, P_w^v) fulfilling the following constraints:

- the edges on $\text{path}_T(v, w)$ are oriented from v to w, and
- the orientation is consistent with c_w.

This definition ensures that the orientation is not conflicting with the realization implied by the coloring c_w and the fixed orientation of $\text{path}_T(v, w)$. The entry $S(w, v, c_w)$ is defined analogously with the difference that here we assume that the edges on $\text{path}_T(v, w)$ are oriented from w to v. Note that the cross pairs having only one endpoint in T_w^v are not contained in P_w^v, and hence do not contribute to the weight of an orientation of (T_w^v, P_w^v). Observe that $\max_{c_r \in \mathcal{C}_r} S(r, r, c_r)$ is the maximum weight of an orientation of the whole instance

(T, P, ω) since $T_r^r = T$ and we build the maximum over all colorings of the cross pairs in Q_r. Next, we provide a strategy to compute the entries of S in accordance with this definition. In the update step, we need to adjust the tables of a vertex w with the tables of its children. Doing so, we have to ensure that we only consider colorings that are not contradictory to each other. Let c_u denote a coloring of Q_u and c_w denote a coloring of Q_w. We use $c_u|c_w$ to denote that c_u and c_w agree in the coloring of the cross pairs in $Q_u \cap Q_w$, that is, for all $q \in Q_u \cap Q_w$ it holds that $c_u(q) = c_w(q)$. Finally, let $W_{LCA}(w, c_w)$ denote the sum of the weights of the cross pairs $[s, t] \in Q_w$ with $c_w([s, t]) = 1$ that have w as their least common ancestor.

For the computation of S, visit each vertex $w \in V$ in a post-order traversal of T. For each w consider all vertices $v \in V$ with $v \prec w$ or $v = w$. Moreover, let $u_1, \cdots, u_\ell$ denote the children of w (if w is not a leaf). Then, for each coloring $c_w \in \mathcal{C}_w$, set $S(v, w, c_w) := -\infty$ if c_w is infeasible for case (1), otherwise set (omit the sum if w is a leaf)

$$S(v, w, c_w) := A(v, w) + W_{LCA}(w, c_w) + \sum_{i=1}^{\ell} M(u_i, v, w, c_w) \quad (8)$$

where

$$M(u_i, v, w, c_w) := max\{ max\{S(v, u_i, c_{u_i}) - A(v, w) : c_{u_i} \in \mathcal{C}_{u_i}, c_{u_i}|c_w\}, \; max\{S(u_i, w, c_{u_i}) : c_{u_i} \in \mathcal{C}_{u_i}, c_{u_i}|c_w\}\} \quad (9)$$

and $S(w, v, c_w) := -\infty$ if c_w is infeasible for case (2), otherwise set (omit the sum if w is a leaf)

$$S(w, v, c_w) := A(w, v) + W_{LCA}(w, c_w) + \sum_{i=1}^{\ell} M(u_i, w, v, c_w) \quad (10)$$

where

$$M(u_i, w, v, c_w) := \max\{ \max\{S(u_i, v, c_{u_i}) - A(w, v) : c_{u_i} \in \mathcal{C}_{u_i}, c_{u_i}|c_w\}, \; \max\{S(w, u_i, c_{u_i}) : c_{u_i} \in \mathcal{C}_{u_i}, c_{u_i}|c_w\}\}. \quad (11)$$

Herein, A is defined exactly as in the proof of Theorem 2.

Correctness. For the correctness, we argue that for v, $w \in V$ with $v \prec w$ and for a coloring $c_w \in \mathcal{C}_w$ that is feasible for case (1), the maximum weight of an orientation of (T_w^v, P_w^v) consistent with c_w that orients the edges on $path_T(v, w)$ from v to w is $A(v, w) + W_{LCA}(w, c_w) + \sum_{i=1}^{\ell} M(u_i, v, w, c_w)$, and, hence, $S(v, w, c_w)$ is computed correctly. To this end, first consider the case that w is a leaf. Then, T_w^v is identical to $path_T(v, w)$ and $Q_w = \emptyset$. Hence, in this case exactly the source–target pairs $[s, t] \in P_w^v$ with $s \prec t$ are satisfied whose total weight per definition is $A(v, w)$. Second, consider the case that w is an internal vertex with children $u_1, \cdots, u_\ell$. Moreover, let $\vec{T}_w^v$ be an optimal orientation consistent with c_w that orients the edges on $path_T(v, w)$ from v to w. Assume that this orientation contains for a child u_i the arc (u_i, w). Then, with respect to w and u_i, the subgraph of $\vec{T}_w^v$ induced by the vertices in $V_{u_i}^w$ is an orientation of $T_{u_i}^w$ consistent with a coloring c_{u_i} that clearly agrees with c_w. Thus, the contribution of u_i to the weight of $\vec{T}_w^v$ is the maximum $S(u_i, w, c_{u_i})$ over all $c_{u_i} \in Q_{u_i}$ that agree with c_w. Similarly, if $\vec{T}_w^v$ contains the arc (w, u_i) for a child u_i, the subgraph of $\vec{T}_w^v$ induced by the vertices in $V_{u_i}^v$ is an orientation of $T_{u_i}^v$ consistent with a coloring c_{u_i} that clearly agrees with c_w. Thus, the contribution of the pairs in P_w^v with at least one endpoint in T_{u_i} is the maximum of $S(v, u_i, c_{u_i}) - A(v, w)$ over all $c_{u_i} \in \mathcal{C}_{u_i}$ that agree with c_w (here, we have to subtract the number of satisfied pairs with both endpoints on $path_T(v, w)$ that are already accounted for by the term $A(v, w)$ in (8)). Finally, observe that the contribution of the cross pairs q in P_w^v with $c_w(q) = 1$ for which w is the least common ancestor are not taken into account in the contributions of the u_i's. This is done by the term W_{LCA} in (8). The argumentation for the correctness of the computation of $S(w, v, c_w)$ follows analogously.

Running time. Next, we analyze the running time. We use the following notation and implementation details. For $w \in V$ let $Q_w = \{p_1^w, \ldots, p_{n_w}^w\}$. A coloring $c_w : Q_w \to \{0, 1\}$ is realized by a tuple $(c_1, \ldots, c_{n_w}) \in \{0, 1\}^{n_w}$ with $c_w(p_i^w) = c_i$ for all $1 \le i \le n_w$. Moreover, the dynamic programming table S is realized by two tables $S_{v,w}^{up}$ and $S_{v,w}^{down}$ for every pair v, $w \in V$ with $v \prec w$ or $v = w$ with an entry for every coloring $c \in \{0, 1\}^{n_w}$ where $S_{v,w}^{up}(c) = S(w, v, c)$ and $S_{v,w}^{down}(c) = S(v, w, c)$. The table A is computed exactly as in the proof of Theorem 2 in $O(n^2)$ time in a preprocessing step. Moreover, note that after $O(n)$ preprocessing time, least common ancestors of the source–target pairs can be found in constant time [12].

To prove the running time, we show that for every pair v, w with $v \prec w$ or $v = w$ the computation in (8) and (10) can be done in $O(2^{q_v} \cdot q_v \cdot \deg_T(w))$ time. We focus on the computation of (8). The running time analysis for (10) follows by the same arguments. The crucial observation is that the summands in the sum in (8) are independent of each other in the sense that the determination of the maximum for one child u_i (the computation of $M(u_i, v, w, c_w)$) does not depend on the decision made for a different child. Hence, for the computation of the entries of $S_{v,w}^{down}$ proceed as follows. Consider each child u of w one

after another. Let $Q_w = \{q_1, \ldots, q_s, q'_1, \ldots, q'_x\}$ and $Q_u = \{q_1, \ldots, q_s, q''_1, \ldots, q''_y\}$, that is, $\{q_1, \cdots, q_s\} = Q_w \cap Q_u$. The crucial point is that we assume that the tables $S^{up}_{w,u}$ and $S^{down}_{v,u}$ are sorted in lexicographical order of the colorings $\{0, 1\}^{n_u}$. This ensures that the colorings of u that agree with a coloring $c_w \in \{0, 1\}^{n_w}$ are ordered consecutively in $S^{up}_{w,u}$ and $S^{down}_{v,u}$. Since the tables $S^{up}_{w,u}$ and $S^{down}_{v,u}$ contain each at most 2^{q_v} entries, the sorting can be achieved in time $O(2^{q_v})$ using bucket sort. Then, for each fixed v, w, and u all the values $M(v, w, u, c)$ can be computed in $O(2^{q_v} q_v)$ time in one iteration over $S^{up}_{w,u}$ and $S^{down}_{v,u}$ and, hence, for all children of w the running time is bounded by $O(2^{q_v} \cdot q_v \cdot \deg_T(w))$. Thus, the overall running time is bounded by

$$O\left(\sum_{v,w \in V} 2^{q_v} \cdot q_v \cdot \deg_T(w)\right) = O(2^{q_v} q_v n^2), \tag{12}$$

since $O(\sum_{w \in V} \deg_T(w)) = O(n)$ in trees. □

Bounded Signal Flow Over Edges

Let m_e be the maximum number of paths that pass through an edge. We consider MTO instances where m_e is limited. We show that the problem is linear-time solvable for $m_e \leq 2$, but NP-hard for $m_e \geq 3$, thereby establishing a dichotomy on the complexity of MTO with respect to m_e.

For the polynomial-time algorithm, we employ the following lemma.

Lemma 1. *If $m_e \leq 2$, then the treewidth of $G_c(T, P)$ is at most two.*

Proof. We make use of the following characterization of graphs of treewidth at most two [21]. A graph has treewidth at most two if it can be reduced to the empty graph by the exhaustive application of the following data reduction rules:

(1) deleting vertices of degree 0 or 1,

(2) deleting a degree-2 vertex whose two neighbors are adjacent, and

(3) adding an edge between the two neighbors of a degree-2 vertex v if the neighbors are non-adjacent, and subsequently deleting v.

We show that if $m_e \leq 2$, then in the conflict graph $G_c(T, P)$ we can find a vertex to which one of the above rules applies. Further, we show that the modified smaller conflict graph is still a conflict graph of some MTO instance. Thus, the claim follows by induction.

Clearly, if Rule (1) or (2) applies to a vertex v in the conflict graph, then we can just delete the corresponding pair in the MTO instance, and the conflict graph of the resulting MTO instance is identical to the graph that results by deleting v.

Next, we show that if neither Rule (1) nor Rule (2) applies, then we can find a vertex v in the conflict graph to which Rule (3) applies. To this end, let (T, P) with $T = (V, E)$ denote an MTO instance and assume that T is rooted at an arbitrarily chosen inner vertex r. Moreover, among all vertices that are the least common ancestors of a pair in P, let x be one with maximum distance to the root r (that is, a deepest least common ancestor). We distinguish two cases based on whether x is an endpoint of a pair with both endpoints in T_x.

First, consider the case that x is the endpoint of a pair $[s, t] \in P$ with $s, t \in V(T_x)$. Let y be the child of x that is contained in the path between s and t. Observe that by the choice of x there is no pair with both endpoints in T_y. Hence, for every pair that is in conflict with $[s, t]$, the corresponding path contains the edge $\{x, y\}$. Thus, since $m_e = 2$, the pair $[s, t]$ is in conflict with at most one other pair, and therefore the corresponding vertex has degree at most one in the conflict graph: a contradiction to the fact that neither Rule (1) nor (2) apply.

Second, consider the case that x is not an endpoint of any pair with both endpoints in $V(T_x)$. Moreover, let $p = [s, t] \in P$ be an arbitrarily chosen pair with $s, t \in V(T_x)$. Let y_1 and y_2 denote the two children of x such that (without loss of generality) $s \in V(T_{y_1})$ and $t \in V(T_{y_2})$. Let $v_{[s,t]}$ denote the vertex of G_c corresponding to $[s, t]$. First, note that by the assumption that Rule (1) does not apply, $v_{[s,t]}$ has degree at least two. Moreover, by the choice of x there is no pair with both endpoints in $V(T_{y_1})$ or in $V(T_{y_2})$. Thus, every pair that is in conflict with $[s, t]$ uses either the edge $\{x, y_1\}$ or the edge $\{x, y_2\}$. Thus, since $m_e \leq 2$ and $\deg_{G_c}(v_{[s,t]}) \geq 2$, there are exactly two pairs $p' = [s', t']$, $p'' = [s'', t''] \in P$ that are in conflict with $[s, t]$. Assume without loss of generality that $t' \in V(T_{y_1})$ and $s'' \in V(T_{y_2})$(see Figure 1 for an illustration). Since Rule (2) does not apply to $v_{[s,t]}$, we can assume that p' and p'' are not in conflict

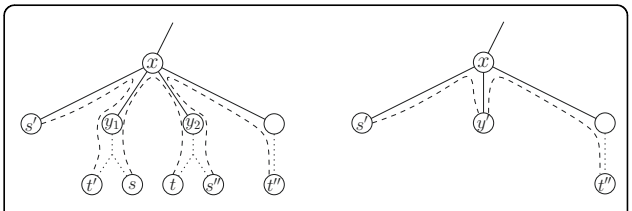

Figure 1 Applicability of reduction. Left: Illustration of the structure for a deepest least common ancestor x of the pairs as described in the proof of Lemma 1. The dashed lines represent the paths between the endpoints of a pair. If $[s, t]$ is in conflict with two other pairs, then these pairs must be conflicting with $[s, t]$ in the edges $\{x, y_1\}$ and $\{x, y_2\}$ since x a deepest least common ancestor. Right: Illustration of the replacement of the subtrees rooted at y_1 and y_2 by a single vertex y'. Note that (s', y') and (y', t'') are in conflict.

with each other. Hence, Rule (3) can be applied to $v_{[s,t]}$. Let G'_c denote the graph that results by first making the two neighbors of v adjacent and subsequently deleting v. It remains to show how to transform the MTO instance such that the conflict graph of the new instance is identical to G'_c. To this end, consider the MTO instance that results by deleting the vertices in $V(T_{\gamma_1}) \cup V(T_{\gamma_2})$, removing the pairs $[s, t]$, $[s', t']$, and $[s'', t'']$ and subsequently adding a vertex y', making y' adjacent to x, and adding the pairs $[s', y']$ and $[y', t'']$ (see Figure 1 for an illustration). Clearly, $[s', y']$ and $[y', t'']$ are in conflict. Moreover, since only the pairs p, p', and p'' have endpoints in $V(T_{\gamma_1}) \cup V(T_{\gamma_2})$, this transformation does not change the conflicts with the other pairs. Further, we have that $m_e \leq 2$ in the resulting MTO instance.

Since width-two tree decompositions can be constructed in linear time [21] and weighted Vertex Cover can be solved in linear time on graphs with constant treewidth [15], this yields linear-time solvability for Weighted Maximum Tree Orientation with $m_e \leq 2$.

Theorem 4. *If $m_e \leq 2$, then* Weighted Maximum Tree Orientation *can be solved in linear time.*

Proof. To be able to determine the path between a pair $[s, t]$ in $O(n)$ time, we root the tree arbitrarily and calculate in linear time a data structure that allows least common ancestor queries in constant time [12]; the path can then be found by going upwards from s and t until hitting their least common ancestor, and then joining the two partial paths. We then construct the conflict graph by marking for each path the corresponding edges with the pair and the direction, and then registering a possible conflict for each tree edge. Since there can be only linearly many markings and conflicts, the construction takes $O(n)$ time. A tree decomposition of width two can then be found in linear time [21], and, as mentioned above, solving weighted Vertex Cover on a graph with treewidth at most two takes only linear time, too [15]. □

We can further prove that for $m_e \geq 3$, MTO is NP-hard even on stars, that is, on trees where all leaves are attached to the same vertex. The proof is by reduction from MaxDiCut.

Theorem 5. Maximum Tree Orientation *on stars with $m_e \geq 3$ is NP-complete.*

Proof. As Medvedovsky et al. [3] pointed out, the NP-hard MaxDiCut problem, defined as follows, can be reduced to MTO on stars.

> MaxDiCut
> Given a directed graph $G = (V, A)$ and a nonnegative integer k, is it possible to find a subset of vertices $C \subseteq V$ such that there are at least $|A| - k$ arcs $(v, w) \in A$ with $v \in C$ and $w \notin C$?

From a MaxDiCut instance $(G = (V, A), k)$, one constructs an equivalent MTO instance $(T = (V', E), P, k)$ by setting $V' := V \cup \{r\}$, $E := \{\{v, r\} \mid v \in V\}$, and $P := A$, where $r \notin V$ is a new root vertex [3]. Clearly, if a MaxDiCut instance has maximum degree three, then it reduces to an MTO instance with $m_e \leq 3$. Thus, it remains to show that MaxDiCut with maximum degree three is NP-hard. (Unfortunately, there seems to be no apt reduction from the undirected version MaxCut, which is NP-hard for maximum degree three [22].)

MaxDiCut can also be formulated as the problem to delete up to k arcs to obtain a graph where every vertex is only startpoint or only endpoint of arcs. We can characterize such graphs by a forbidden substructure consisting of three vertices u, v, w connected by the arcs (u, v) and (v, w) (the arcs (u, w) and (w, u) may or may not be present). Thus, if we ignore graphs with multiple arcs between two vertices, we have three forbidden induced subgraphs on three vertices. In this way, MaxDiCut is similar to the Transitivity Deletion problem [23], which given a directed graph, asks for up to k arc deletions to make it *transitive*, that is, to fulfill for all u, v, $w \in V$ that $(u, v) \in A \wedge (v, w) \in A \Rightarrow (u, w) \in A$. Transitive graphs are characterized by two of the three forbidden subgraphs for MaxDiCut; the subgraph with $\{(u, v), (v, w), (u, w)\} \subseteq A$ is not forbidden. However, if we examine the directed graphs that are produced in the reduction from 3-SAT that proves NP-hardness of Transitivity Deletion [23, Sect. 3.1], we notice that this substructure does not occur, and cannot be created by arc deletions. Thus, solving Transitivity Deletion and MaxDiCut on these directed graphs is equivalent. Since the constructed instances also have degree at most three, we obtain the NP-hardness of MaxDiCut with maximum degree three. It is easy to see that MTO is contained in NP, so we obtain the claimed theorem. □

Observations on Protein Interaction Networks

The goal in this section is to explore the space of practically meaningful parameterizations, here focusing on biological applications. We first performed experiments based on the same data as used by Medvedovsky et al. [3]. The network is a yeast protein-protein interaction network from the Database of Interacting Proteins (DIP) [24], containing 4,737 vertices and 15,147 edges. The cause–effect pairs were obtained from gene knockout experiments by Yeang et al. [2] and contain 14,502 pairs. After discarding small connected components and contracting cycles, we obtained a tree with 1,278 vertices and 5,569 pairs. (These numbers differ slightly from the ones stated by Medvedovsky et al. [3]. We do not use the additional kinase-substrate data, which is only meaningful to evaluate the orientations obtained, and which

requires an arbitrary parameter choice not documented by Medvedovsky et al. [3].)

The resulting tree is, as already observed by Medvedovsky et al. [3], very star-like: there is one vertex of degree 1,151 and 1,048 degree-one vertices attached to it. The remaining 229 vertices have degree 1 to 4. All paths connecting cause-effect pairs pass through the central vertex.

We first note that this MTO instance is actually fairly easy to solve exactly. The Integer Linear Program (ILP) by Medvedovsky et al. [3, Sect. 3.1] and Vertex Cover on the conflict graph solved by either an ILP or a simple branching strategy with data reduction all solve the instance in less than a second. More precisely, the running times are 0.09 s, 0.02 s, and 0.13 s, respectively, on a 2.67 GHz Intel Xeon W3520 machine, using GLPK 4.44 for the ILPs, and with the branching strategy implemented in Objective Caml. The branching strategy finds a vertex v of maximum degree and branches into the two cases of taking v into the vertex cover or taking all neighbors of v into the vertex cover. Before each branch, degree-1 vertices are eliminated by taking their neighbor into the vertex cover. The search in the second branch is cut short when the accumulated vertex cover is larger than that of the first branch.

Note that all three algorithms do not require the parameter k (number of unsatisfied pairs) as input, but will determine the minimum k such that there is a solution.

The reason that these strategies work so well is probably due to the low value of the parameter k: only 77 cause-effect pairs cannot be satisfied. This limits the size of the branch-and-bound tree that underlies all three methods.

In Table 1, we examine several other parameters. Since there are still p_t = 5,569 pairs left after contracting all cycles in the network, using this parameter for a fixed-parameter algorithm seems infeasible. Unfortunately, since all paths run through a single vertex, the parameter m_v is not any more useful. Only about 5% of the pairs are cross pairs after the data reduction, so q is already a more promising parameter. However, with a value of q = 417, this parameter seems not very helpful. Even if we eliminate pairs that do not conflict with any other pairs, leaving only n_c = 1,287 pairs, we still find at least 306 cross pairs (parameter q'). Again, because all paths run through a single vertex, considering cross pairs per vertex does not help here. In summary, for this particular instance the number of unsatisfiable pairs k is clearly the most useful parameter.

To examine the effect of the sparseness of the input instance on the various parameters, we investigated another yeast protein interaction network assembled by Nir Yosef from various sources (see references in [25]). In this network, each edge is annotated with a probability of interaction. Thus, by thresholding, we can obtain graphs of different sparseness. The results are shown in Table 2.

Table 1 Network parameters

Parameter		Value
n	Number of network vertices	4,654
m	Number of network edges	15,104
p	Number of pairs	14,155
n_t	Vertices in MTO instance	1,278
p_t	Number of pairs in MTO instance	5,569
n^*	Number of vertices in star	1,049
m_v	Max. number of pairs per vertex	5,569
m_e	Max. number of pairs per edge	371
q	Number of cross pairs	417
q_v	Max. number of cross pairs per vertex	417
q'	Number of cross pairs after data reduction	306
q'_v	Max. number of cross pairs per vertex after data reduction	306
n_c	Number of vertices in conflict graph	1,287
m_c	Number of edges in conflict graph	4,626
k	Number of unsatisfiable pairs	77

Values for various parameters for the protein interaction network instance from Medvedovsky et al. [3].

We see that, here, the parameter k is not always a clear winner. When the network becomes sparser, the components that will be shrunk to a single vertex by the cycle contraction will be smaller, leaving fewer pairs with both endpoints on the same tree vertex, and thereby increasing the number of potential conflicts. Only for very high thresholds, the parameter becomes small again, since then the original instance is already much smaller. Still, all instances can be solved in less than one second by the three algorithms mentioned above, which exploit low values of k.

We also see that for denser graphs, the parameter values based on the number of cross pairs are quite low, e.g. q'_v = 3 for the whole graph. Thus, it seems likely that these instances can be quickly solved by the algorithm from Theorem 3, running in $O(2^{q'_v} \cdot n^2 \cdot q'_v)$ time. One possible explanation for the low value for these parameters is that the networks exhibit a linear structure. For example, if each protein can be assigned a distance to the nucleus, and interactions mostly transport information to or from the nucleus, then we would expect to have only few cross pairs.

The parameter m_v could be expected to be not too high in biological networks, since otherwise this would make the network less robust, since elimination of one vertex would disrupt too many paths. However, one vertex in the tree under consideration can actually correspond to a very large component in the original graph, which weakens this effect. Therefore, this parameter is more useful in sparser graphs, where not too many graph vertices are joined into a tree vertex. However, for the given instances, it seems small enough to be

Table 2 Thresholded network parameters

threshold	n	m	p	n_t	p_t	n^*	m_v	m_e	q	q_v	q'	q'_v	n_c	m_c	k
0.000000	5385	39921	14393	799	2014	750	2014	59	7	7	3	3	115	292	17
0.154420	4530	35041	11522	747	2203	705	2203	298	27	27	20	20	475	1632	40
0.371369	4254	32135	10740	796	2443	749	2443	275	47	47	35	35	528	2424	46
0.573290	3871	27128	9445	777	2225	704	2225	268	32	32	13	13	140	311	32
0.573313	2546	8977	5279	638	2311	477	2310	208	252	252	151	151	561	2394	68
0.830093	2206	7136	4346	643	2206	449	2206	192	304	304	193	193	727	4017	83
0.886308	1407	3646	1607	441	787	260	785	45	106	106	88	88	311	1876	75
0.943001	1135	3069	920	361	464	195	463	32	57	57	42	42	179	801	44
0.954421	1039	2504	843	350	489	175	461	45	85	73	71	61	215	3001	81
0.957338	895	2060	681	304	405	119	375	39	64	54	58	50	240	3092	89
0.965986	874	2018	666	299	477	103	411	165	90	78	85	75	358	12284	110
0.984753	668	1676	312	206	163	95	162	20	7	7	6	6	55	222	15
0.989212	581	1322	188	192	167	69	161	86	24	24	24	24	141	1088	32
0.989233	307	681	71	121	70	32	66	36	21	21	11	11	52	219	7
0.990409	294	666	28	114	27	26	26	21	2	2	2	2	9	8	2

Parameters for the largest connected component of the protein interaction network assembled by Nir Yosef [25] with different thresholds for the edge probability. The uneven gaps in the sizes of the instances are because many edges have identical weights.

exploited only for fairly small instances, where other parameters would give good results, too.

The parameter m_e could similarly be expected to be low in sparse networks; however, the NP-hardness result already for $m_e \geq 3$ (Theorem 5) makes practical use of this parameter unlikely.

Conclusions

We started a parameterized complexity analysis of (Weighted) Maximum Tree Orientation, obtaining a more fine-grained view on the computational complexity of this NP-hard problem. In this line, there are still several challenges for future investigations. For instance, it is open whether MTO is fixed-parameter tractable with respect to the parameter "number of satisfied pairs" (n - k). Further, in the spirit of "distance-from-triviality parameterization" [19,20] it would be interesting to study the parameterized complexity of MTO with respect to the parameter "number of all possible pairs minus the number of input pairs"–recall that for parameter value zero MTO is polynomial-time solvable [11]. MTO restricted to stars is still NP-hard, but then at least one quarter of all input pairs can always be satisfied [3]. Hence, it would be interesting to study above guarantee parameterization [15,20] with respect to the number of satisfied pairs. MTO can be translated into a vertex covering problem (see Proposition 1) on a graph class that is K_4-free–this motivates to study whether vertex covering on this graph class can be done faster than on general graphs. Clearly, MTO brings along numerous further parameters and parameter combinations which can make a more comprehensive multivariate complexity analysis [20] very attractive. Often, it is desirable to not only list a single solution, but to enumerate all optimal solutions. Our dynamic-programming-based algorithms seem suitable for this. Following Gamzu et al. [8] and extending the studies for MTO as pursued here to the more general case of mixed graphs with partially already oriented edges is of high interest. First steps in this direction have very recently been undertaken by Silverbush et al. [9] and Elberfeld et al. [26]. Finally, it seems promising to examine the parameters based on cross pairs in other networks such as communication networks, and to try to exploit these parameters for other hard network problems.

Acknowledgements

A preliminary version of this work appeared in the proceedings of the 1st International ICST Conference on Theory and Practice of Algorithms in (Computer) Systems (TAPAS '11), volume 6595 in Lecture Notes in Computer Science, pages 104-115, Springer 2011.

JU and partly FH were supported by the Deutsche Forschungsgemeinschaft (DFG), research project PABI (NI 369/7).

Major parts of the work were done while BD and DK were with the Universität Tübingen, FH was with the Humboldt-Universität zu Berlin, and RN and JU were with the Friedrich-Schiller-Universität Jena. We are grateful to two anonymous referees whose insightful remarks helped to improve the presentation of our work.

Author details

[1]Fakultät für Mathematik und Wirtschaftswissenschaften, Universität Ulm, Ulm, Germany. [2]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany. [3]Institut für Theoretische Informatik, Universität Ulm, Ulm, Germany.

Authors' contributions

All authors contributed more or less equally, RN initiating the study of MTO under the viewpoint of multivariate complexity analysis and JU coming up with the major algorithmic ideas which have been worked out in more detail by DK. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 22 March 2011 Accepted: 25 August 2011
Published: 25 August 2011

References

1. Werther M, Seitz H, Eds: In *Protein-protein interaction. Volume 110.* Advances in Biochemical Engineering/Biotechnology. Springer; 2008.
2. Yeang CH, Ideker T, Jaakkola T: **Physical network models.** *Journal of Computational Biology* 2004, **11(2-3)**:243-262.
3. Medvedovsky A, Bafna V, Zwick U, *et al*: **An algorithm for orienting graphs based on cause-effect pairs and its applications to orienting protein networks.** In *Proc 8th WABI. Volume 5251.* LNBI, Springer; 2008:222-232.
4. Karzanov AV: **Èkonomnyj algoritm nahoždeniâ bikomponent grafa [in Russian: An efficient algorithm for finding the bicomponents of a graph].** *Trudy tret'ej zimnej školy po matematičeskomû programmirovaniu i smežnym voprosam [Proceedings of the 3rd Winter School on Mathematical Programming and Related Problems]* Moscow Engineering and Construction Institute (MISI); 1970, 343-347.
5. Tarjan RE: **Depth-first search and linear graph algorithms.** *SIAM Journal on Computing* 1972, **1(2)**:146-160.
6. Alm E, Arkin AP: **Biological networks.** *Current Opinion in Structural Biology* 2003, **13(2)**:193-202.
7. Sharan R, Ideker T: **Modeling cellular machinery through biological network comparison.** *Nature Biotechnology* 2006, **24**:427-433.
8. Gamzu I, Segev D, Sharan R: **Improved orientations of physical networks.** In *Proc 10th WABI. Volume 6293.* LNBI, Springer; 2010:215-225.
9. Silverbush D, Elberfeld M, Sharan R: **Optimally orienting physical networks.** In *Proc 15th RECOMB. Volume 6577.* LNBI, Springer; 2011:424-436.
10. Gitter A, Klein-Seetharaman J, Gupta A, *et al*: **Discovering pathways by orienting edges in protein interaction networks.** *Nucleic Acids Research* 2011, **39(4)**:e22.
11. Hakimi SL, Schmeichel EF, Young NE: **Orienting graphs to optimize reachability.** *Information Processing Letters* 1997, **63(5)**:229-235.
12. Harel D, Tarjan RE: **Fast algorithms for finding nearest common ancestors.** *SIAM Journal on Computing* 1984, **13(2)**:338-355.
13. Downey RG, Fellows MR: *Parameterized Complexity* Springer; 1999.
14. Flum J, Grohe M: *Parameterized Complexity Theory* Springer; 2006.
15. Niedermeier R: **Invitation to Fixed-Parameter Algorithms.** No. 31 in Oxford Lecture Series in Mathematics and Its Applications, Oxford University Press; 2006.
16. Niedermeier R, Rossmanith P: **On efficient fixed-parameter algorithms for weighted vertex cover.** *Journal of Algorithms* 2003, **47(2)**:63-77.
17. Song Y, Liu C, Huang X, *et al*: **Efficient parameterized algorithms for biopolymer structure–sequence alignment.** *IEEE/ACM Trans Comput Biology Bioinform* 2006, **3(4)**:423-432.
18. Bodlaender HL: **A partial k-arboretum of graphs with bounded treewidth.** *Theoretical Computer Science* 1998, **209(1-2)**:1-45.
19. Guo J, Hüffner F, Niedermeier R: **A structural view on parameterizing problems: distance from triviality.** In *Proc 1st IWPEC. Volume 3162.* LNCS, Springer; 2004:162-173.
20. Niedermeier R: **Reflections on multivariate algorithmics and problem parameterization.** In *Proc 27th STACS. Volume 5.* Leibniz International Proceedings in Informatics, Schloss Dagstuhl - Leibniz-Zentrum für Informatik; 2010:17-32.
21. Arnborg S, Proskurowski A: **Characterization and recognition of partial 3-trees.** *SIAM Journal on Algebraic and Discrete Methods* 1986, **7(2)**:305-314.
22. Yannakakis M: **Edge-deletion problems.** *SIAM Journal on Computing* 1981, **10(2)**:297-309.
23. Weller M, Komusiewicz C, Niedermeier R, *et al*: **On making directed graphs transitive.** *Journal of Computer and System Sciences* 2011.
24. Salwinski L, Miller CS, Smith AJ, *et al*: **The database of interacting proteins: 2004 update.** *Nucleic Acids Research* 2004, , **32 Database:** D449-D451.
25. Bruckner S, Hüffner F, Karp RM, *et al*: **Topology-free querying of protein interaction networks.** *Journal of Computational Biology* 2010, **17(3)**:237-252.
26. Elberfeld M, Segev D, Davidson CR, *et al*: **Approximation algorithms for orienting mixed graphs.** In *Proc 22nd CPM. Volume 6661.* LNCS, Springer; 2011:416-428.

doi:10.1186/1748-7188-6-21
Cite this article as: Dorn *et al.*: Exploiting bounded signal flow for graph orientation based on cause–effect pairs. *Algorithms for Molecular Biology* 2011 **6**:21.